John Munguia

CS 405

July 6, 2025

In this assignment, I modified the add\_numbers() and subtract\_numbers() functions to detect and prevent numeric overflow and underflow across various C++ data types. To accomplish this, I added logic that checks the result before performing each arithmetic operation using std::numeric\_limits<T>::max() and std::numeric\_limits<T>::min(). If an addition or subtraction would exceed the allowable bounds of the data type, the function returns false to indicate failure; otherwise, it performs the operation and returns true, with the result passed back by reference.

The test\_overflow() and test\_underflow() functions were also updated to handle the return value from the modified arithmetic functions. They now print whether an overflow or underflow was detected, or display the correct result if the operation succeeded. This approach prevents silent failure and ensures the program behaves predictably. A challenge in this process was ensuring compatibility with both signed and unsigned types, which was resolved by using type-safe checks and testing across all required primitive types. The final implementation works correctly and provides clear feedback to the user when numeric limits are exceeded.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.